SF 298 MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
12/16/96

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
~~Investigating a Distributed Multiple-Agent for Command and Control of Theatre Wide Conflicts~~ See Report

5. FUNDING NUMBERS
DAAH04-95-1-0364

6. AUTHOR(S)
Wolf Kohn, Anil Nerode

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)
Mathematical Sciences Institute
Cornell University
409 College Avenue, Rm. 321
Ithaca, NY 14850

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
ARO 34801.1-MA-SDI

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

We present a preliminary design of a hardware architecture for computing initial segments of primitive recursive functions and iterative processes. The formulation of the architecture is based in a paradigm which proposes a procedure for (1). encoding a function or process and, (2)carrying out the computation. The paradigm is firmly rooted in the formalism of quantum mechanics. We propose as our representation of the architecture a generic regular multiparticle, two-dimensional lattice. This lattice is a model of crystal structures that in principle, can be produced in the lab today.

19970210 193

14. SUBJECT TERMS

15. NUMBER IF PAGES
22

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

DTIC QUALITY INSPECTED 1

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

QUANTUM WAVE PROCESSOR RESEARCH REPORT

BY

Wolf Kohn
and
Anil Nerode

Quantum Wave Processor Research Report

Wolf Kohn
Anil Nerode

Problem Statement

This document describes ongoing research for the formulation, analysis and implementation of a programmable device, termed Quantum Function Evaluator (QFE), that uses quantum state propagation as a paradigm for computing objects of a class I of functions herein referred to as the computable class. The proposed research is divided into three phases: *Formulation, Evaluation* and *Implementation.* We will discuss an outline of the first phase in the next sections.

Outline of formulation Phase

Over a period of several years, Kohn and Nerode have conducted joint and independent research to explore formal methods in Hybrid System Theories [1], [2], [3], [4], [5], [6], [7] to determine effective computational procedures for generating control laws in a variety of application domains. The quantum computing paradigm and its possible implementation proposed for this research effort, constitute a significant extension of the results in our early efforts. This outline provides a brief synopsis of our proposed paradigm and a preliminary sketch of a possible implementation architecture. The Formulation phase of our study is composed of two major tasks: the detailed specification of the proposed *paradigm* and the detailed specification of an *implementation architecture*. We will outline the major features of these two items next.

Paradigm

The main aspects of our proposed paradigm are summarized in Figure 1. The proposed computational paradigm is composed of seven sequential steps. We outline their functionality next.

Input Function: Each Input Function is either a map of the form:

$$f : S_1 \times \cdots \times S_N \to D$$

where S_i, $i = 1, ..., N$ and D are finite subsets of the naturals of the form:

$$S_i = \{0, \cdots, N_i\}$$
$$D = \{0, \cdots, N_d\}$$

or a solution of an iterative process of the form:

$$y^i_{n+1} = f^i(y^1_n, \ldots, y^k_n),\ i = 1, \ldots, k$$

with

$$f^i : S^k \rightarrow S,\ S = \{0, 1, \ldots, N_S\}$$

Let p be the natural number defined by:

$$p = \max_{i,d}\{\{N_i,\ i = 1, \ldots, N\}, N_d\}$$

Then, we can express f by the *encoding*:

$$F : [0, 1, \ldots, p^{N-1}] \rightarrow [0, 1, \ldots, p]$$

$$\text{For} \quad x = \sum_{s=0}^{N-1} n_s \cdot p^{N-1-s}$$

$$F(x) = \begin{cases} f(n_0, \ldots, n_{N-1}), & \text{If defined} \\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

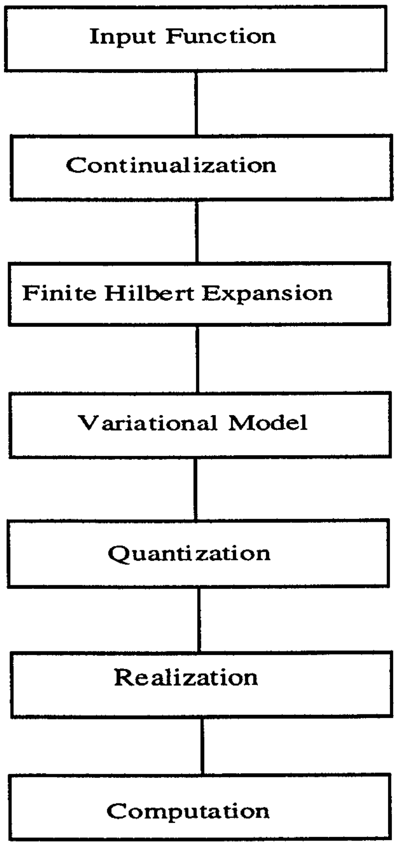


Figure 1. Quantum Computing Paradigm

Thus, the encoding is a discrete function with domain on certain points of the real line, taking values in a subset of points of the real line coinciding with the values in the range of f and mapping to 0 any encoding x that corresponds to a tuple not in the domain of f.

The central objective behind our paradigm is to find effective and fast means to compute discrete finite-valued functions via *Quantum Approximations* to their encoding. We will characterize the nature of these approximations later on. We conclude the overview of the Input Function element of our paradigm with a formal definition of the class I of computable functions.

The class I of computable input functions is defined as follows:

(i) I contains all the functions encodable in one step as above.

(ii) If $\{f_1, \cdots, f_K, f_i : S_1 \times \cdots \times S_N \rightarrow S_{N+1}, \text{K finite}, S_i = \{1, ..., N_i, N_i, \text{finite}\}\}$ is a subset of I, so is their direct sum: $f_1 \oplus \cdots \oplus f_K : S_1 \times \cdots \times S_N \rightarrow S^K_{N+1}$.

(iii) I contains the projection functions:
$p_i : S_1 \times \cdots \times S_N \rightarrow S_i, \; p_i(n_1, ..., n_i, ..., n_N) = n_i$.

(iv) If $\{f_1, \cdots, f_K, f_i : S_1 \times \cdots \times S_N \rightarrow S_{N+1}, \text{K finite}, S_i = \{1, ..., N_i, N_i, \text{finite}\}\}$ is a subset of I and $g : S^K_{N+1} \rightarrow S_{N+1}$ is in I, so is the composition: $g(f_1, \cdots, f_K) : S_1 \times \cdots \times S_N \rightarrow S_{N+1}$.

(v) If for each $n \in \mathrm{N}$, $g : \mathrm{N} \times S_1 \times \cdots \times S_K \rightarrow \mathrm{N}$ is in I so is $\min\{n, g(n, n_1, \cdots, n_K) = 0\}$.

(vi) If $g : \mathrm{N} \times S_K \rightarrow S_K$ is in I, then the family of functions,
$\{f : \mathrm{N} \times S_1 \times \cdots \times S_K \rightarrow S_{K+1}, f(n+1, n_1, ..., n_k) = g(n, f(n, n_1, ..., n_k))\}$ is in I.

(vii) Any function constructed by the finite application of (i)-(vi) is in I.

We note that the defining characteristic of I is that its elements are either an encoding or can be transformed into an encoding via finite number of steps (ii)- (vii) above. We conclude the outline of the input function with the following observation: although the class I of computable functions I includes only completely specified functions, it can be extended to a larger class which include discrete partially specified functions (relations). We will discuss this extension in a future report.

Continualization: We can complete the encoding function F into a *step function* Φ with the following identity:

$$\Phi : [0, p^{N-1}) \rightarrow [0, p^{N-1})$$
$$\Phi(y) = F(x), \quad \text{for each } y, \; y \in [x, x+1), \; x \in \{0, ..., p^{N-1}\} \tag{2}$$

Figure 2 illustrates this identity for the encoding of the function given in Table 1. We refer to the process of constructing Φ as *Continualization*. With some needed modifications, this continualization process can be extended to discrete functions that are specified as solutions of iterations of the form:

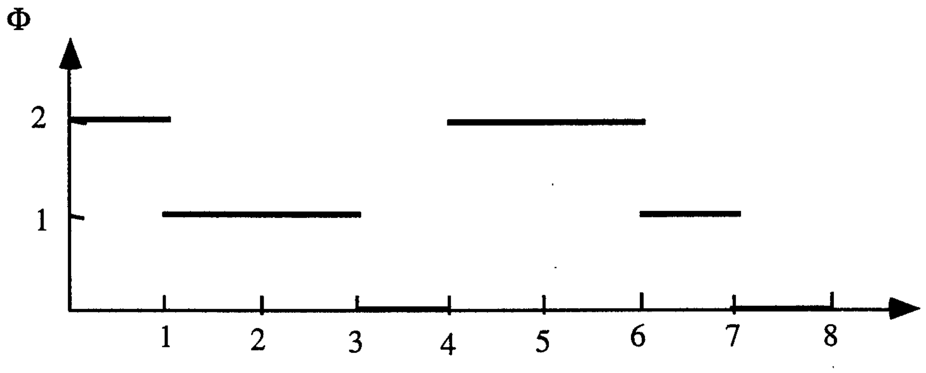
Table 1.

n_0	n_1	f
0	0	2
0	1	1
0	2	1
1	0	0
1	1	2
1	2	2
2	0	1
2	1	0
2	2	0

Original Function

x	F
0	2
1	1
2	1
3	0
4	2
5	2
6	1
7	0
8	0

Encoding



Step Function

Figure 2. Continualization of a function

$$Z_{k+1} = g(Z_k, k)$$
with (3)
$$Z_k \in S^N, \ S = \{0, ..., N, \text{N finite}\} \text{ for each k, k an integer}$$

Following Kushner and Clark [8], the encoding and continualization of a process of the form of (3) leads to a differential equation of the form of (4) together with a sampling rule of the form of (5):

$$\dot{x}(t) = G(x(t), t) + B(t) \tag{4}$$

$$G(\cdot, t) = \text{ continualized version of } g(\cdot, k),\ t \in \left[k\Delta, (k+1)\Delta\right),\ k \in N \tag{5}$$

and B(t) is a 'convergence' function satisfying the following conditions:
B is bounded, and B(t) goes to zero as $t \rightarrow \infty$ quadratically. Under general conditions, one can show [9] that particular solutions of (4) are asymptotically stable in the sense of Lyapunov. In (5), Δ is a scaling parameter chosen to transform the iteration variable k into the time variable t.

Finite Hilbert Expansion: The continualized functions, appearing in (1) or (4) may be expanded in a discrete Hilbert space γ_p in terms of an orthogonal basis set such as the set of p-valued Chrestenton functions. The general form of a Chrestenton basis function is given below.

$$\text{Chrestenton function: } \varphi^j(x) = \exp\left(\frac{2\pi}{p}\, i \sum\nolimits_{s=0}^{N-1} j_{(N-1-s)} \cdot x_s\right)$$
$$\text{with } j = \sum\nolimits_{s=0}^{N-1} j_s \cdot p^{N-1-s},\quad x = \sum\nolimits_{s=0}^{N-1} x_s \cdot p^{N-1-s}$$

For the sake of simplicity, we will continue our discussion here under the assumption that the selected bases are the Chrestenton functions, although during the study of *tunneling* as a mechanism for implementing *quantum oracles* we may need to switch to other bases.

Let $B_p = \left\{\varphi^k \mid \varphi^k : \left[0, p^{N-1}\right] \rightarrow \left[0, p^{N-1}\right],\ k = 0, \cdots, p^N - 1\right\}$ be an orthogonal basis for γ_p. That is, $\int \varphi^k(x) \cdot \overline{\varphi}^l(x) dx = c \cdot \delta_{k,l}$ for each k, l in $\left\{0, \cdots, p^N - 1\right\}$. The function F can be expanded in terms of B_p as follows:

$$\Phi(x) = \sum_{x \in [0,p]^N} \beta_k \cdot \varphi^k(x) \tag{6}$$

with

$$\beta_k = \int_0^{p^N} \Phi(x) \cdot \overline{\varphi}^k(x) \cdot dx \Bigg/ \int_0^{p^N} \varphi^k(x) \cdot \overline{\varphi}^k(x) \cdot dx$$

We note that if we know the spectrum $\{\beta_k\}$ We can easily compute Φ, via (1), extract the encoding F from it and from F determine the original function f. However even for functions of more than academic interest, the number of terms in the spectrum runs in the range of 10^5 to 10^{12}. The central idea, in this regard, is that we can find very close approximations to this evaluation by a *hardware quantum device,* which we will discuss in the next section. For the moment, in this section, we continue with the description of the paradigm.

Variational Model: Given a computational process to be carried out, expressed in the form of the differential equation (4), we want to formulate the computation of the solution x(t), as a variational problem. Our purpose is to proceed from this to obtain a quantum mechanical "program " for computing the solution. For the purposes of this report, we will describe the variational formulation formally. Mathematical rigor will follow in future reports.

The variational formulation is of the form:

$$\min_x \int L(x, \dot{x}, t) \cdot dt \tag{7}$$

where L known, as the lagrangian function, is chosen so that for specific boundary conditions the solutions of (7) and (4) coincide. For the purposes of our procedure we assume that L is three times continuously differentiable in its arguments. We now proceed to describe how L is determined from the function G, constructed earlier. Towards this objective we write the necessary conditions for optimality in (7), The Euler Lagrange conditions. They are expressed by the following second order differential equation:

$$\dot{x}L_{\dot{x}x} + \ddot{x}L_{\dot{x}\dot{x}} + L_{\dot{x}t} - L_x = 0 \tag{8}$$

Here and in the derivations that follow subindices indicate partial derivatives.

Differentiating (4) with respect to t, to obtain a definition for $\ddot{x}$, replacing in (8), and differentiating the result with respect to $\dot{x}$, we obtain, after some algebra, the following partial differential equation :

$$L_{\dot{x}\dot{x}t} + G_x L_{\dot{x}\dot{x}} + \dot{x}L_{\dot{x}\dot{x}x} + (\dot{x}G_x + G_t + B_t)L_{\dot{x}\dot{x}\dot{x}} = 0 \tag{9}$$

Let

$$q(x, \dot{x}, t) = L_{\dot{x}\dot{x}}(x, \dot{x}, t) \tag{10}$$

In terms of q, (9) takes the form,

$$q_t + G_x q + \dot{x}q_x + (\dot{x}G_x + G_t + B_t)q_{\dot{x}} = 0 \tag{11}$$

That is, q satisfies a linear hyperbolic equation, whose solution can be computed by the method of characteristics ([10]). Once a solution for q is obtained L can be determined from (10) by a double quadrature. In summary, given (4) we construct by the procedure sketched above the corresponding lagrangian.

Quantization: In order to quantize the program described by (7), one introduces the canonical conjugate variables: generalized position and momentum [12]. In the canonical Quantization these are chosen as follows: position variable, representing the position operator, is represented by x, and the momentum variable is chosen as:

$$p = L_{\dot{x}} \tag{12}$$

The components of x, p satisfy the canonical commutation relations:

$$[x_k, p_l] = i\delta_{k,l} \quad k, l = 1, \dots, N \tag{13}$$

Next, we define the Hamiltonian of the system as follows:

$$H(x, p) = \sum p_k \cdot \dot{x}_k - L \tag{14}$$

The equation of motion of the system representing the program to be executed is then the Schödinger equation given by :

$$i\hbar \frac{\partial}{\partial t} |\Psi(t)\rangle = H\left(x, \frac{1}{i}\frac{\partial}{\partial x}\right) |\Psi(t)\rangle \tag{15}$$

Realization: Assume that a physical system, the hardware, is available to us. This system is characterized by the Hamiltonian operator H_0. The realization step consists in *engineering* a field Hamiltonian H_f such that the hardware, when interacting with this field satisfies the following condition:

$$|H_0 + H_f - H| \leq \varepsilon \tag{16}$$

where $|\cdot|$ denotes a suitably selected operator norm and ε is an engineering parameter chosen to satisfy precision requirements for the computation.

The general idea is that the *discrete spectrum* of the composite Hamiltonian $H_0 + H_f$ approximates the spectrum of the function or process being computed (see (6) above). Precisely, we will devote our next report to prove the following result:

"*Let*

$\{\lambda_k,\ k = 1, \dots, N\}$ be a subset of the eigenvalues of $H_0 + H_f$ with corresponding eigenvectors $\{\Psi_k\ k = 1, \dots, N\}$:

$$(H_0 + H_f)|\Psi_k\rangle = \lambda_k|\Psi_k\rangle$$

then

$$|\lambda_k - \beta_k| \leq \varepsilon/N$$

and

$$\int_0^{\infty} \left|\varphi^k(x) - \Psi_k(x)\right| \cdot dx \leq \varepsilon$$

where φ^k is a Chrestenton function"

Thus computing a discrete function or a discrete process can be approximated by exciting the hardware system appropriately and reading the resulting spectrum.

Computation: The computation, to a large extent, consists in simulating the system characterized by $H_0 + H_f$ using the hardware characterized by H_0, and excited by the field whose Hamiltonian is H_f. Our research about computation is geared towards developing an effective translator that converts an input function or process of the class I into a Hamiltonian operator H and a resolution element that extracts the excitation Hamiltonian H_f. Our long term objectives are to design and implement a prototype system that operates according to the paradigm outlined in this section.

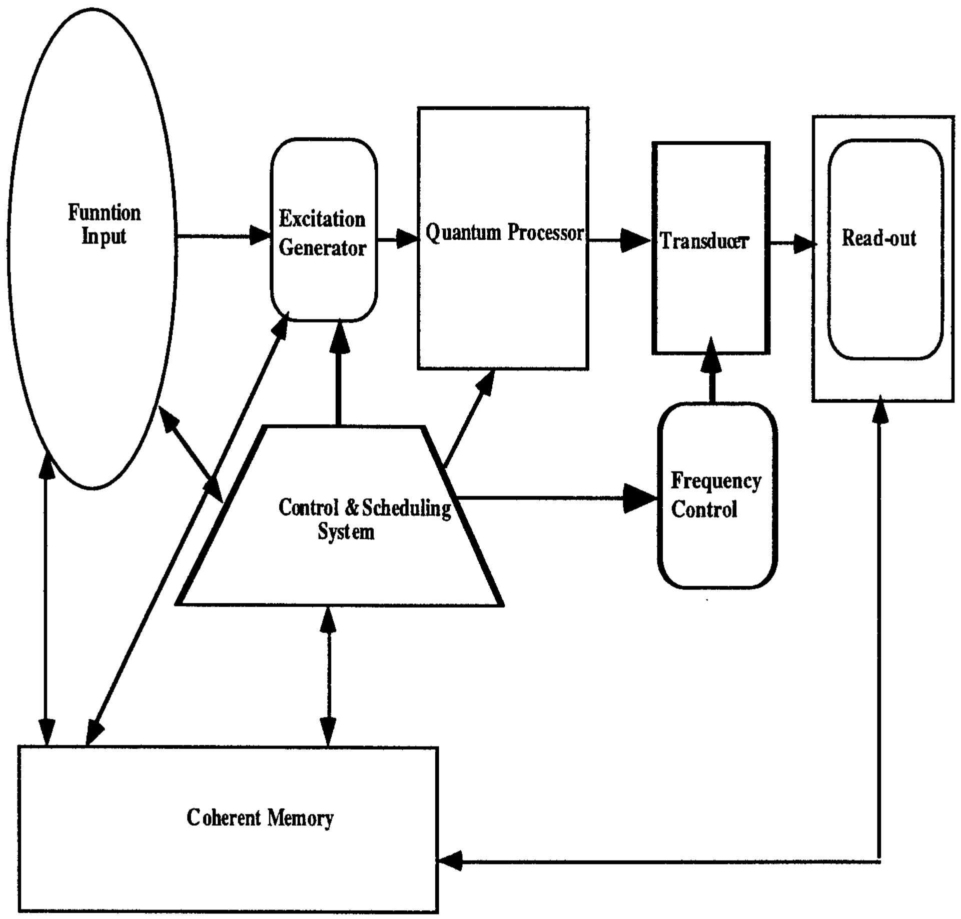


Figure 3. Functional Architecture for Quantum Computing

Architecture

In this section we propose a preliminary design of an architecture that implements the paradigm presented in the previous section. A functional model of our proposed architecture is depicted in Figure 3. We will next discuss briefly the functionality of the elements composing our proposed architecture:

Input Function - Given a discrete function or process to be computed, this element implements the *compilation process;* that is, it determines the excitation Hamiltonian, H_f, which encodes the function or process . This is a symbolic computation whose steps are carried out following the paradigm described in the previous section. These steps can be described by an iterative process such as (3) so in *principle* the compilation process could be carried out by the architecture itself. We will explore the mechanization of the compilation process in this direction.

Excitation Generator - This is a device that realizes *physically* the field excitation encoded in H_f. That is, the excitation generator converts a description of the computation into a physical implementation . The idea is for this device to *address* each individual particle of the quantum process described below and to excite them according to the desired state behavior dictated by:

$$i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle = \left(H_0 + H_f\right)\left(x, \frac{1}{i}\frac{\partial}{\partial x}\right)|\Psi(t)\rangle$$

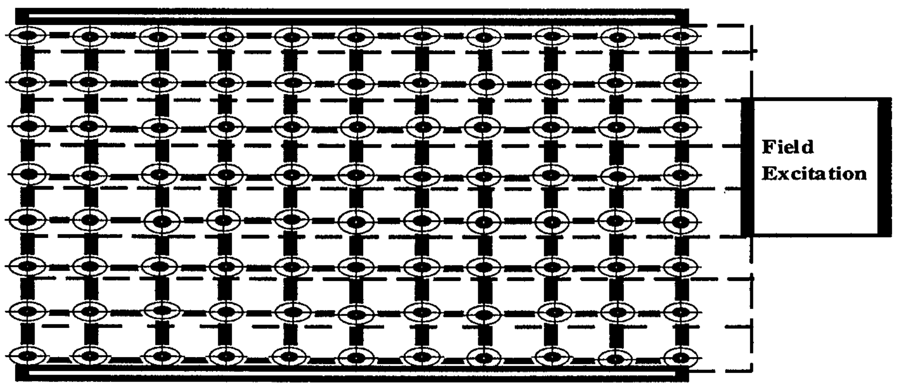
Our research in this area will be focused on the development of a mechanization process for realizing this functionality. We will consider two alternatives: optical or particle (electron) excitation. A more detailed discussion of these two alternatives will be provided in our next report.

Quantum Processor - This is the device which actually carries the computation. Without excitation, it is a realization of the Hamiltonian H_0. After excitation it is a realization of $H = H_0 + H_f$. We will provide next an abstracted model of the physical characteristics of the Quantum Processor.

The Quantum Processor is an array of identical *particles* assembled into a regular lattice. For the purposes of discussion we will consider a two-dimensional lattice. A later report will be devoted to the formulation of the physical characteristics of the lattice; in this report we will be concerned only with the formulation of some of its computational behavior.

A diagram of the Quantum Processor is shown in Figure 4. The device is composed of two elements: the computational lattice of particles and the field excitation device. From a computational point of view each particle which is allocated to a node in the lattice, can be represented as a non-deterministic, two-level state automaton, and an interface function called Input selector function as shown in Figure 5. We proceed to describe their functionality next.

In the *Lower Automaton*, the block labeled 'State transition' in figure 5 characterizes the programmable discrete spectrum of the particle. The states of the automaton represent energy levels, and edges represent allowable energy transitions. This is illustrated in Figure 6 below.



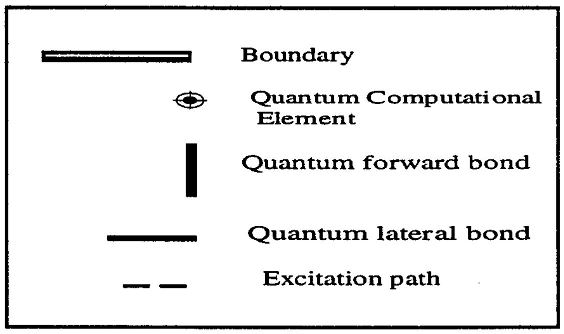


Figure 4. Quantum Processor

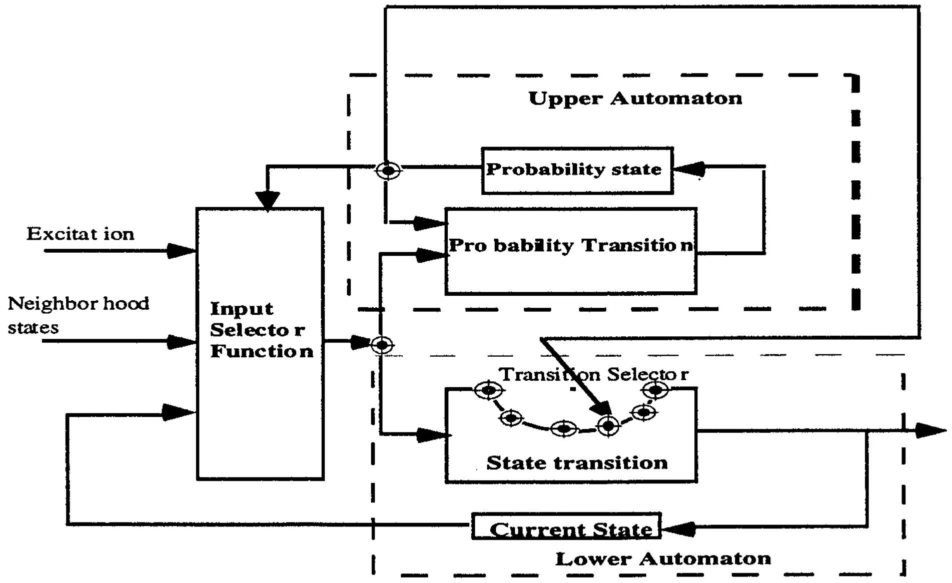


Figure 5. Quantum Particle

In Figure 6, the energy levels correspond to eigenvalues (or sets of eigenvalues forming a band grouped together as a single eigenstate) of the corresponding Hamiltonian. The edges correspond to energy transitions. Edges pointing up are driven by excitation: that is, the transition is effected by absorption of energy from the excitation. Edges pointing down correspond to relaxation effects: the particle releases energy of the appropriate frequency either to neighborhood particles or to the field (see Figure 7).

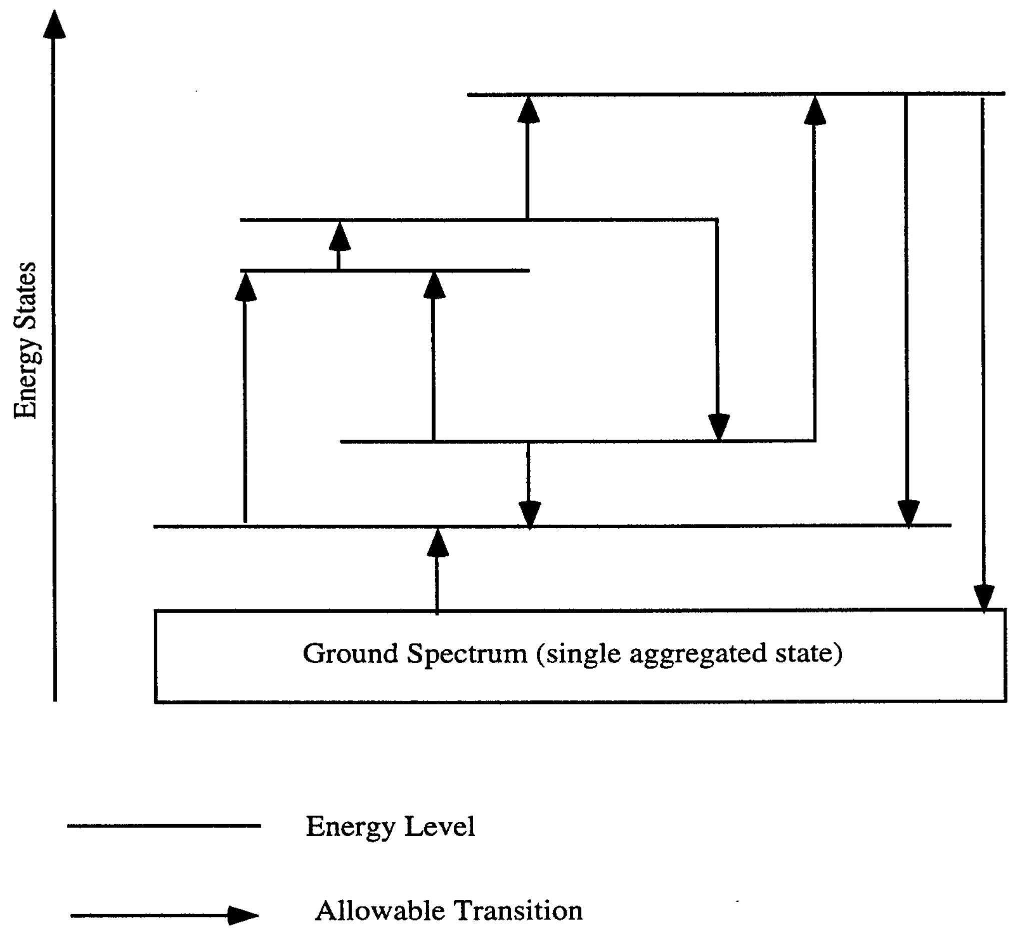


Figure 6. State energy transition example

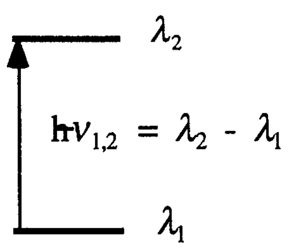


Figure 7. A state transition

In Figure 7, λ_1 and λ_2 are energy levels, and the transition between them is driven by excitation energy to frequency $\nu_{1,2}$. Note from Figure 5 that energy can come to the particle either by interaction with the other particles or from the external excitation. Relaxation transitions are similar.

The state transition in the lower level automaton is controlled by the *mixing state probability density transition* computed by the *Upper Level Automaton*. State transition in this automaton is called 'probability transition' in Figure 5.

The mechanism that implements commands issued by the upper level automaton in the lower level automaton is called *Transition Selector* in Figure 5. We will explain its functionality next. For this we need some preliminary definitions.

Let $S = \{|\Psi_k(t)\rangle,\ k \in N_p,\ N_p \text{ finite}\}$ be the states of the lower automaton in each particle. the let Δ be the *update time* of the particle. The update time of the particle is the determined to be larger than 10 times the maximum relaxation time of any of the state transitions in the lower automaton. The state of the lower level automaton in each interval $[t, t+\Delta)$ is a *Chattering Combination* of the elements of S. A Chattering Combination of the set of functions in the interval is a function $|\Psi(t)\rangle$ defined as follows:

$$|\Psi(\tau)\rangle = \begin{cases} \Psi_{i_1}(\tau) & \tau \in I_{i_1}(t) \\ \Psi_{i_2}(\tau) & \tau \in I_{i_2}(t) \\ \vdots & \\ \Psi_{i_{N-1}}(\tau) & \tau \in I_{i_{N-1}}(t) \\ \Psi_{i_N}(\tau) & \tau \in I_{i_N}(t) \end{cases} \tag{17}$$

where $I_{j_k}(t),\ j_k \in N_p,$ is a semi-open interval in $[t, t+\Delta)$:

$$I_{j_k}(t) = \left[t + \sum_{l=0}^{k-1} \Delta_{j_l}(t),\ t + \sum_{l=0}^{k} \Delta_{j_l}(t) \right) \tag{18}$$

We note from (17) that the composite state $|\Psi(\tau)\rangle$ is constructed by 'stitching' together the pure states in S. The time spent in pure state $|\Psi_{j_k}\rangle \in S$, $\Delta_{j_k}(t)$ is a function of the characteristic excitation or relaxation times associated with the state [11]. We note that

$$\sum_k \Delta_{j_k}(t) = \Delta \tag{19}$$

and, if we define

$$\alpha_{j_k}(t) = \frac{\Delta_{j_k}(t)}{\Delta} \tag{20}$$

then equation (19) can be written as

$$\sum_k \alpha_{j_k}(t) = 1 \tag{21}$$

The set of all Chattering Combinations of S in the interval $[t, t + \Delta)$ is denoted by $\hat{S}(t, t+\Delta)$. The union of these sets for all time is denoted by $\hat{S}$. Now we can specify the operation of the Transition Selector in the lower automaton of our particle model. For each interval $[t, t + \Delta)$, the Transition Selector receives a command from the upper level automaton, which consists of an ordered tuple of coefficients $\langle \alpha_{j_k}(t), j_k \in N_p \rangle$ satisfying (21), and then it computes the mixed state of the particle, $|\Psi(t)\rangle$, according to (17). The state of the lattice is composed of the states of each of its particles. We will see shortly that the Chattering Coefficients have the interpretation of state occupancy probabilities. To demonstrate that, we need to discuss the dynamic structure of the upper level automaton.

In general, there is not enough information to say that the lattice or any of its particles is characterized by a specific state function. The best we can do, in order to describe the computation, is to give a probabilistic description. In the quantum formalism, this description is referred to as the probability density description [12].

Let S be the set of primitive states of the lower level automaton. Let p_j be the probability that the particle is in state $\Psi_j \in S$. The probability density operator [13], ρ, is defined by

$$\rho(t) = \sum_j p_j |\Psi_j(t)\rangle\langle\Psi_j(t)| \tag{22}$$

By differentiating (22) and using (15), after some algebra, we obtain an (operator) differential equation for ρ :

$$i\hbar \frac{\partial}{\partial t}\rho = H \cdot \rho - \rho \cdot H \tag{23}$$

The operator is termed the commutator and is written as $[H, \rho]$. Thus

$$i\hbar \frac{\partial}{\partial t}\rho = [H, \rho] \tag{24}$$

Given an observable characterized by, say, operator C, the associated observed quantity, denoted by $\langle C \rangle$, is given by the expectation of C relative to ρ :

$$\langle C \rangle = \mathrm{trace}(\rho \cdot C) \tag{25}$$

Equation (24) characterizes the computation carried out by the upper level automaton. Equation (25) characterizes the Transducer of our architecture (see Figure 3). We will devote a future report to discussion of this device in detail.

Notice that the state function, computed by the lower level automaton, could be given as a linear combination of the states in S. We chose to model it as a Chattering Combination, which turns out to be equivalent in a specific sense (as we will show shortly) because in this form it will allow us to formulate the *sequence* of excitation steps (realized by the excitation element, see Figures 4 and 5). To a large extent, programming the quantum processor is tantamount to determining this sequence. To justify this statement, we must explain the sense in which the Chattering and Linear State Combinations are equivalent, because an extension of this result to Excitation Hamiltonians will provide us with a strategy for implementing excitation sequencing .

The equivalence between Linear and Chattering Combinations of state functions from a given set S is established in the following version of the Chattering Lemma [14]:

Chattering Lemma. Let $S = \left\{ |\Psi_k(t)\rangle,\ k \in N_p,\ N_p \text{ finite} \right\}$ and let $\hat{S}$ be the set of chattering combinations of S. Let ε, ε real and positive be given. There exist state functions $|\Theta_j\rangle \in \hat{S}$, defined for each tuple $\left\{ \alpha_1, \ldots, \alpha_{n_p} \middle| \alpha_l \geq 0,\ \sum_{l=1}^{n_p} \alpha_l = 1 \right\}$, such that

$$\max_t \left| \int_0^t \left\{ |\Theta_j(\tau)\rangle - \sum_{l=1}^{n_p} \alpha_l |\Psi_l(\tau)\rangle \right\} \cdot d\tau \right| < \varepsilon \qquad (26)$$

for all $\left(\alpha_1, \ldots, \alpha_{n_p}\right)$.

The proof of this lemma, while not difficult, requires extensive manipulations. We will provide it in a companion report devoted exclusively to the chattering aspects of our proposed architecture. Notice that the lemma says that every chattering combination of state functions of the form of (17) on a set S, can be realized as a linear combination of elements from S with an arbitrary small error in the integral sense (see (26)). We also have the fact that under strong continuity assumptions on the state functions the converse of the lemma is also true. By choosing the boundary conditions in our lattice model appropriately, this assumption is not limiting .

Now we proceed to describe the functionality of the *Input selector function.* This function models the interaction of the particle with the excitation and with the other particles in the lattice. Some examples of possible particle interaction are shown in Figure 8. For simplicity, only nearest neighboring interactions are shown, but the model is not limited to these cases. The central task implemented by the input selector function is to establish on one hand the interaction of a particle with the other particles in the lattice and on the other hand the interaction of a particle with the excitation field in order to characterize these two tasks we look more closely at the excitation field Hamiltonian H_f.

Locally, centered in a particle, it is convenient to write H_f as the sum of two terms, the first, H_e, corresponding to the interaction of the particle with the excitation, and the second H_l corresponding to the interaction of the particle with its internal energy and with the interaction energies of the other particles in the lattice. Thus we write

$$H_f = H_e + H_l \qquad (27)$$

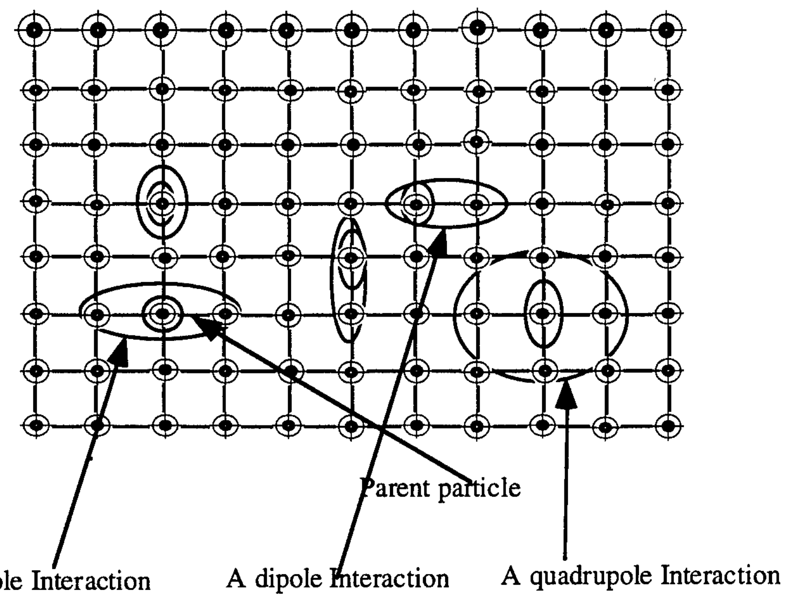


Figure 8. Some examples of particle interaction

Under general assumptions about the lattice, the interaction Hamiltonian H_I for given particle can be written as

$$H_I|\Psi\rangle = g_\Psi\left(H_I^{i_1},\dots,H_I^{i_k}\right)|\Psi\rangle \tag{28}$$

where $H_I^{i_j}$ for each j is the interaction Hamiltonian of the given particle with its 'neighboring' particle i_j, and g_Ψ is the neighborhood function at the current state of the particle, Ψ. The neighborhood function represents the local (at the current state Ψ) structure of interaction of the particles in the lattice. This structure can be determined by building the lagrangian associated with the lattice and go to the procedure of quantization that we discussed earlier. We will carry out this task once the details of the physics of the lattice are defined in the evaluation phase. The approach consist in defining *potentials* to characterize interaction. For example, a dipole (particle-to-particle) interaction between a particle located in coordinates x, y of the lattice may be characterized by a potential V of the form:

$$V(x,y,t) = d_{x-y}(t)\frac{1}{|x-y|}$$

where d_{x-y} is a relaxation function.

For the purposes of analysis and also to determine a detailed formulation of the realization step in our paradigm, it is convenient to assume that a finite set of primitive excitation Hamiltonians $E = \left\{H_e^i,\ i = 1,\dots,n_e\right\}$ can be realized and that implementation of our excitation is carried out by *chattering* among the elements of E over the update interval Δ. Specifically,

$$H_e|\Psi(\tau)\rangle = \begin{cases} H_e^{i_1}|\Psi(\tau)\rangle & \tau \in I_{i_1}(t) \\ H_e^{i_2}|\Psi(\tau)\rangle & \tau \in I_{i_2}(t) \\ \vdots & \\ H_e^{i_{n-1}}|\Psi(\tau)\rangle & \tau \in I_{i_{n-1}}(t) \\ H_e^{i_n}|\Psi(\tau)\rangle & \tau \in I_{i_n}(t) \end{cases} \tag{29}$$

Where the sets $I_{i_j}(t)$ are defined by expression (18). This type of *probabilistic resonance* is central to our proposed implementation of the quantum processor. The idea is to induce a probability distribution ρ on the states of the particles on the lattice so that the realization criterion is satisfied.

Thus a computation in the lattice is a propagating probabilistic wave-train in which the state of each particle is the probability distribution of its pure states. This is illustrated in Figure 9. Specifically the lattice is at an initial probabilistic state, the programmed excitation is impinged and after a transient period ξ has elapsed, the read out period, the transducer is activated to effect the eigen-value observation. After the observation has been made, the computation process is complete. If the results are not satisfactory the computation is started again from the initial probabilistic state and the read-out period is extended to $\xi_1 > \xi$. This extension period cannot be extended arbitrarily because the thermal relaxation mechanisms in the lattice will induce eventually *decoherence,* that is, the loss of the probabilistic resonance described above.

Let $\{E_i\}$ be the discrete spectrum of the physical Hamiltonian H_0. Then it can be shown using (24), that the transition probability $\rho_{i,j}$ from eigenstate i to eigenstate j satisfies,

$$i\hbar\frac{\partial}{\partial t}\rho_{i,j}(t) = (E_i - E_j)\cdot \rho_{ij} + [H_f, \rho]_{ij} \tag{30}$$

In [11], it is shown that if the system is at state $|\Psi_j\rangle$ at time t= 0, then the presence of the relaxation Hamiltonian (28) causes the corresponding probability density term ρ_{jj} *to decay exponentially with time*. For small values of t, ρ_{jj} is the largest term in the matrix representation of the probability density operator. Assuming no excitation, and using (30) this term satisfies the following equation:

$$i\hbar\frac{\partial}{\partial t}\rho_{jj} = \sum_k\left((H_1)_{jk}\rho_{kj} - \rho_{jk}(H_1)_{kj}\right) \tag{31}$$

with

$$i\hbar\frac{\partial}{\partial t}\rho_{ji} \cong (E_j - E_i)\cdot \rho_{ji} - \rho_{jj}\cdot(H_1)_{ji} \quad \text{for } i \neq j \tag{32}$$

The solution of (31) and (32) for ρ_{jj} is given by:

$$\rho_{jj}(t) = e^{-t/\xi} \tag{33}$$

where ξ is a relaxation time that depends on the physical characteristics of the lattice. Thus the effect of the inter particle excitation Hamiltonian is to randomize the state transitions in each particle. The computation described above will be corrupted by this effect. Therefore, the read-out period ζ must be chosen so that this effect is acceptable; this means $\zeta \ll \xi$. We will use this bound in our design specifications of the quantum processor and also, in our computability analysis.

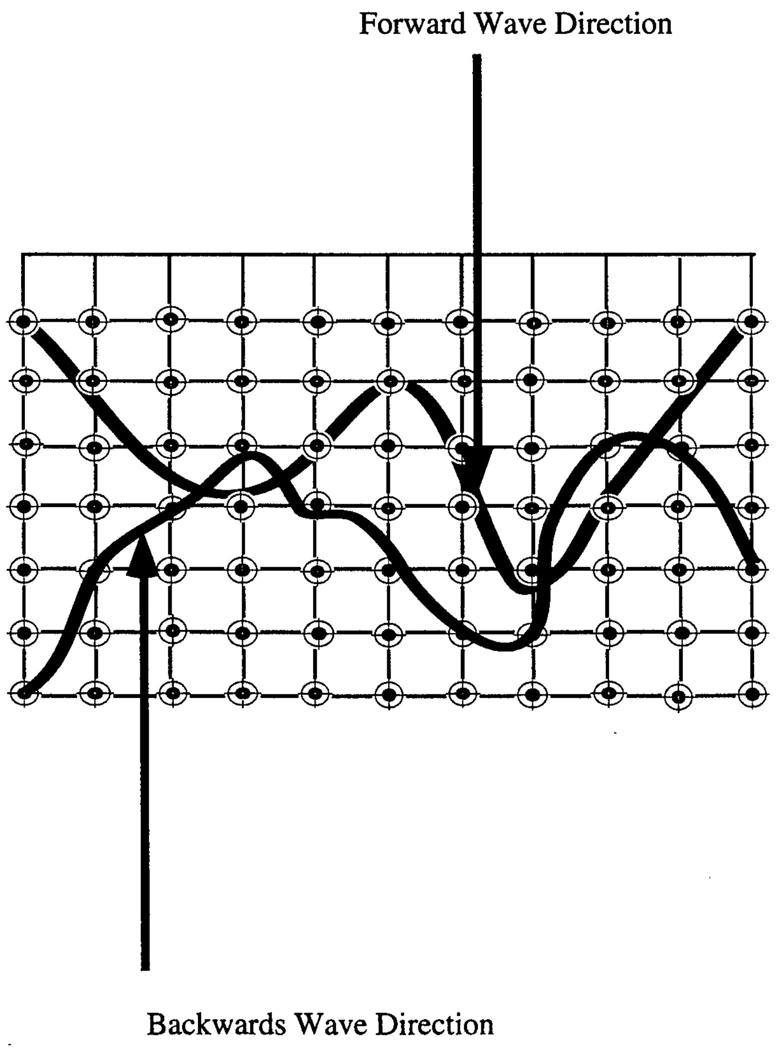


Figure 9. Wave Propagation

Conclusions

In this paper we present a preliminary design of a hardware architecture for computing initial segments of primitive recursive functions and iterative processes. The formulation of the architecture is based in a paradigm which proposes a procedure for 1- encoding a function or process and 2- Carry out the computation. The paradigm is firmly rooted in the formalism of quantum mechanics. We propose as our representation of the architecture a generic regular multiparticle, two-dimensional lattice. This lattice is a model of crystal structures that in principle, can be produced in the lab today.

References

[1] Kohn W. "Distributed Hierarchical Automata with some Applications to Genetics in Prokaryotes", Ph.D. dissertation, MIT EE., 1978, Cambridge Mass.

[2] Kohn W. " The Rational Tree Machine" SPIE vol. 1095 VII pp. 264-274, 1989.

[3] Kohn W. Graham R.V., Butler J.W., "The rational Tree Machine", Proc. IEEE Northcon 88, pp. 330-347, Seattle Oct. 1988.

[4] Kohn W., Kumar S. "Parallel Simulation of the Rational Machine" Boeing Electronics BE.-499-5 July 7, 1989.

[5] W. Kohn and A. Nerode, ``Models for hybrid systems: automata, topologies, controllability and observability", Hybrid Systems ,(R. Grossman, A. Nerode, A. Ravn, H. Rischel Eds.) Lecture Notes in Computer Science 736, Springer-Verlag, 1993, 317-356.

[6] A. Nerode, and W. Kohn ``Multiple agent autonomous hybrid control systems" , in Logical Methods: A Symposium in honor of Anil Nerode's 60th Birthday, (J. N. Crossley, Jeffrey B. Remmel, Richard A., Shore, Moss E. Sweedler, eds.), Birkhauser, 1993.

[7] Wolf Kohn, Anil Nerode, J. B. Remmel, and Xiolin Ge, ``Multiple Agent Hybrid Control:`` Carrier Manifolds and Chattering Approximations to Optimal Control", CDC94.

[8] Kushner H. J. and D. S. Clark "Stochastic Approximation Methods for Constained and Unconstrained Systems", Springer -Verlag NY. 1978.

[9] KUsner H. J. ""general convergence results for stochastic approximations via weak convergence theory" J.Math. Anal.and Applic., 61, 1977, pp. 490-503.